Multiple Algorithms and the Bridge CA Concept

Introduction

The Bridge CA concept [TWG-98-29], [TWG-98-33] has been accepted as the overall approach to building a Federal PKI out of a number of separate agency application specific or agency wide PKIs. The Bridge CA (BCA) approach also provides a path to connect the Federal PKI into the overall national and global PKI. The BCA concept has been incorporated into the FPKI CONOPS document [TWG-98-31] as well. However, the present CONOPS draft does not deal with multiple digital single algorithms in the context of the BCA. An earlier document [TWG-98-18] described an approach to multiple algorithms in a Federal PKI, however it did not specifically consider the effect of the BCA.

Terms

In this paper we use the following terms:

Authority Revocation List (ARL): An indirect CRL that lists all the revoked CA certificates for all the CAs in the FPKI.

Bridge CA (BCA): A CA that is to be a bridge of trust that provide trust paths between the various trust domains of the Federal PKI, as well as between the Federal PKI and non-federal trust domains;

Certificate: A digitally signed document that binds two or more attributes together. In this paper we are only concerned with x.509 digital signature certificates that bind a subject's digital signature public key (as opposed to his key management or encryption key) to his name.

Certificate Revocation List (CRL): A signed list of certificates that have been revoked;

Certification Authority (CA): A trusted entity that issues (i.e., signs and publishes) certificates and/or CRLs;

certification path: a sequence of certificates beginning with a self-signed signature certificate issued by a CA trusted by a relying party and ending with an end-entity's signature certificate, where the issuer of any certificate in the sequence is the subject of the preceding certificate;

end-entity: a certificate holder that is not acting as a CA. In most cases an end user with a certificate.

mixed algorithm certificate: A certificate where the subject's algorithm for the certified key is different from the algorithm used by the issuing CA to sign the certificate.

Principal CA (PCA): A CA within a trust domain that cross-certifies with the Federal BCA. Each trust domain has one principal CA.

relying party: An entity that validates a digital signature;

self-signed certificate: A certificate signed with the key it certifies. It is used by a CA to state (but not authenticate) its public key;

single algorithm certificate: a certificate where the same algorithm is used for the public key certified in the certificate and to sign the certificate;

trust domain: In the Federal context a trust domain is a portion of the Federal PKI that operates under the management of a single *policy management authority*. One or more Certification Authorities exist within the trust domain.

Discussion

Several digital signature algorithms will be used in the government and elsewhere and individual trust domains may standardize on different algorithms. Two algorithms, DSA and RSA are now in common use in the Federal Government, and ECDSA will probably see growth use in the future. The design of the FPKI must be general enough to accommodate these algorithms and must also allow for the introduction of new algorithms. It must provide a way for certificate holders of different algorithms to interoperate.

In principle, any digital signature algorithm can be used to sign a certificate that certifies the key for any public key algorithm. By the terminology defined above, a certificate signed by one algorithm, for a key of a different algorithm, is called a mixed algorithm certificate. The conclusion of [TWG-98-18] and earlier TWG studies on multiple algorithm interoperability was that the best approach is an "end-entity" approach, where:

- the end-entity normally signs with a single algorithm, minimizing the number of keys and certificates he is required to hold and manage;
- in principle, any digital signature algorithm can be used to sign a certificate that certifies any key for any other algorithm;
- if the end-entity needs to interoperate with end-entities who use other algorithms, then his client should be able to validate signatures for other algorithms (but not necessarily to sign with multiple algorithms);
- therefore, in the interest of broad interoperability, Federal users should be encouraged to employ clients that can validate all the common (or FIPS approved) digital signature algorithms;
- End-entity certificates should never be mixed algorithm certificates, since any relying party must be able to validate 2 different algorithms to validate a signature signed under that mixed algorithm certificate (even if the relying party uses the same CA as the signatory), and that plainly increases the likelihood of algorithm interoperability problems. Moreover mixed algorithm certificates are likely to require parameters to be carried in the certificates, which increases their size substantially, and end-entity certificates are by far the most numerous class of certificates.

Although the end-entity approach is the recommendation for the Federal PKI, some clients may not ever implement the ability to validate signatures for all the algorithms used in the Federal PKI. This will surely be common whenever a new signature algorithm is coming into general use. Moreover, other clients may implement only one digital signature algorithm. Therefore, the Federal PKI should be designed, as far as practicable, to ensure that two end-entities who use the same signature algorithm, should be able to find a certification path that does not require use of another algorithm. It is therefore desirable, in the interest of local interoperability, for individual trust domains to use a single signature algorithm. This will minimize the number of certificates that need to explicitly state parameters in the certificates, and minimize the chances for algorithm induced interoperability problems within a trust domain. It will also be desirable for there to be trust paths between domains that use the same algorithm, that require only that one algorithm.

Similarly, whenever two end entities use different signature algorithms, it is desirable that there be a certification path between the two that does not require use of a third signature algorithm. This will maximize the chance that the two can validate the trust paths.

If the CA Alice trusts signs with algorithm "white" and Bob's key is for algorithm "gray," then, if Alice is to validate Bob's signature, there must exist a mixed algorithm certificate for a gray algorithm key, signed with a white algorithm key, somewhere in the certification path between them.

The principle that local trust domains should minimize the use of multiple algorithms within the domain implies that the needed mixed algorithm certificates should be associated with the BCA that connects trust domains. That is, the mixed algorithm certificates that are needed to provide trust paths between users of different algorithms are properly either issued by or to the BCA.

Possible Solutions

Figure 1 illustrates four possible approaches to multiple algorithms with the BCA, for the case of

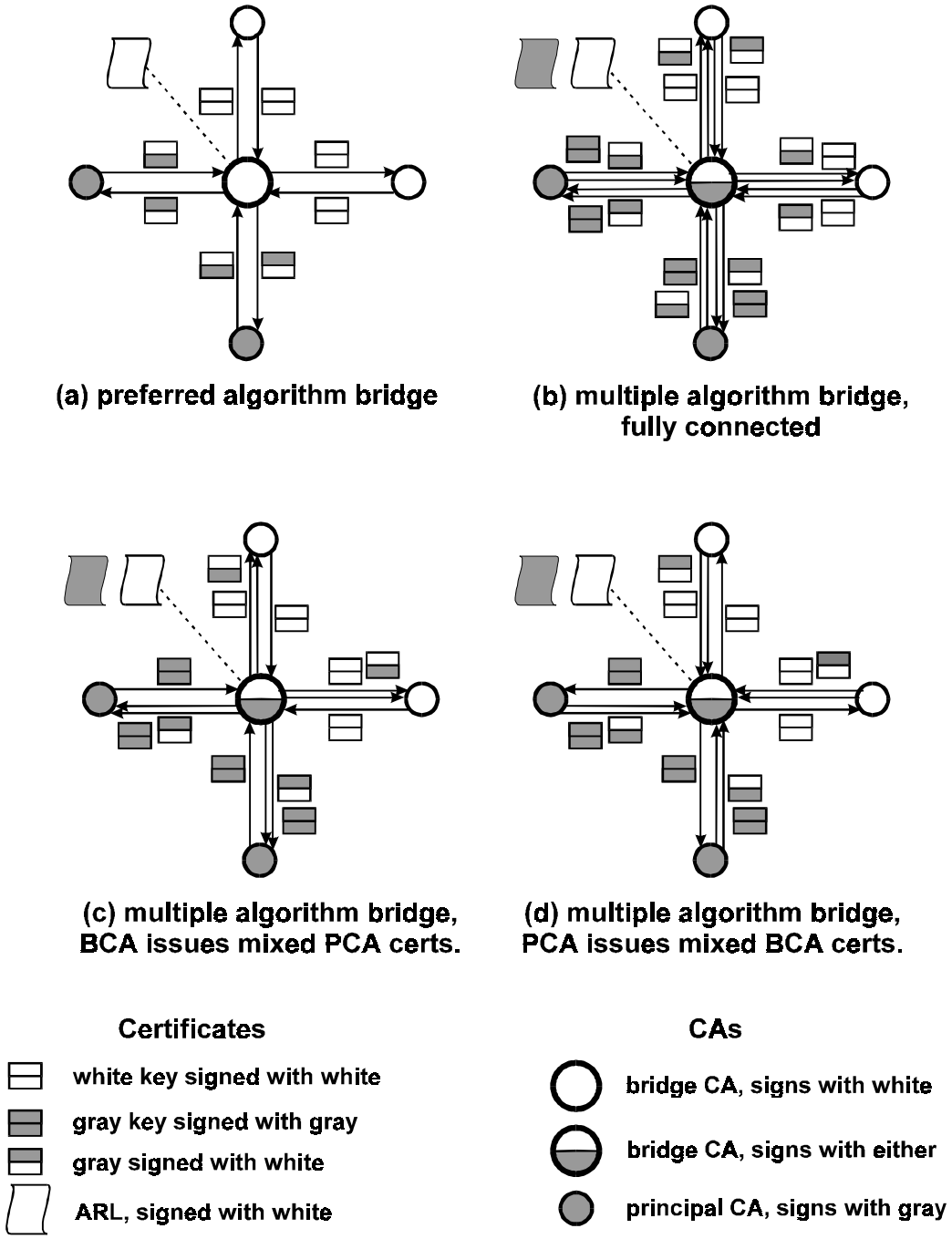

Figure 1 - Single Bridge CA Multi-Algorithm Alternatives

two algorithms, labeled "white" and "gray." Showing only 2 algorithms helps to keep the illustration simple, but we must bear in mind that we will probably ultimately need to accommodate more than two algorithms, and the approach we adopt must consider this.

Preferred Algorithm Approach

One obvious approach, illustrated in Figure 1(a), is to have a single "preferred" algorithm (in the figure "white") for the bridge CA. Every client must be able to validate at least this algorithm if it is to use certification paths created by the bridge CA.

This immediately violates the precept that two users who use the same algorithm should not have to use another algorithm to find a certification path. But it may not be a serious problem if there truly is agreement about the preferred ("white") algorithm choice. Moreover, domains that use the "gray" algorithm might cross certify directly with each other, or there could be a "helper" bridge CA for the Gray algorithm, that just connected gray domains. This might provide a workaround for applications that are only able to implement a single algorithm, and have no need to connect to users of other algorithms.

The obvious difficulty is in picking the preferred algorithm. To paraphrase Orwell, "All algorithms are equal, but some algorithms are more equal than others." If we can agree on a single Bridge CA algorithm, and everyone will implement at least the ability to validate signatures that use it, this is the most efficient choice. Those who don't implement verification of the chosen algorithm will be left out. However, it is not clear how we can select a single preferred algorithm for the Bridge CA, when we cannot manage to do so for more general use. It does seem here that any algorithm to be preferred for this purpose should be one that is efficiently verified, which argues for RSA, I believe.

Another obvious issue with this approach arises when it is time to change the preferred algorithm. It is easy enough, in principle, to replace a BCA with another BCA that uses a different algorithm, but, before we can do that, we must get clients into the hands of all users that can validate the new preferred algorithm.

Multiple Algorithm Bridge CA Approaches

If we cannot select a single preferred algorithm for the Bridge CA, then the Bridge must support multiple algorithms. There are at least 3 somewhat different general approaches to supporting multiple algorithms in the bridge:

- one BCA signs with several algorithms;
- there are several BCAs, one for each algorithm;
- one BCA signs a single PCA certificate with several algorithms.

Single BCA with Multiple Algorithms

In these cases a single BCA signs PCA certificates with several algorithms. There are at least three somewhat different variations.

1. Figure 1 (b) illustrates a "fully connected" multiple algorithm BCA. In this approach the BCA supports several algorithms, and each uses one of these algorithms. For each algorithm the BCA supports it issues a separate certificate for each PCA's key. One will be a single algorithm certificate and the rest will be mixed algorithm certificates. Similarly, each PCA issues a

certificate to the BCA for each of the BCA's keys; one will be a single algorithm certificate and the rest mixed algorithm certificates. This approach generates more BCA-PCA certificates than are strictly necessary to ensure that trust paths exist. Do the redundant certificates offer compensating advantages in better support for some modes of certification path building?

2. Figure 1 (c) illustrates a multiple algorithm BCA, in a PKI where only the BCA issues mixed algorithm certificates. PCAs certify only the BCA key that matches their own algorithm. This provides full trust path connectivity, in the sense that there exists a valid trust path between all the PCAs. It eliminates a number "redundant" certificates as opposed to case (1) above. It does not require PCAs to issue mixed algorithm certificates. If there are n PCA's and m algorithms, the total number of hybrid certificates are $n \times (m - 1)$, and these are in addition to the certificates that would be required for the Preferred Algorithm approach.

3. Figure 1 (d) illustrates a multiple algorithm BCA, in a PKI where only the PCA's issue mixed algorithm certificates to the BCA. Again, this provides full trust path connectivity. It is equivelent to case (2) above except that it is the PCAs that issue mixed algorithm certificates to the BCA, rather than the reverse. It requires the same number of mixed certificates as case (2) above. It has the possible disadvantage that it requires PCAs to issue mixed algorithm certificates[1].

Is there a significant advantage to one of the three cases? Are there certification path building strategies that work better with one or the other? One factor may be lie in the certificate matching rules: **certificateMatch** includes **subjectPublicKeyAlgID**, but nothing for the issuer algorithm. I originally conjectured that this rule can be used to some useful effect in case (3) above, but not in case (2), because, where there are multiple certificates between the BCA and PCA, they all certify the same key in case (2). I haven't really come up with a certification path search strategy that seems to take much advantage of this. But in case (2), PCAs only certify the BCA with single algorithm certificates, so we could use **subjectPublicKeyAlgID** to narrow the search to all PCAs that use the relying party's algorithm, if we are constructing the path from the signatory to the relying party (which may be the usual way to do it).

Multiple Bridge CA Approaches

There is a strong correlation between the first three multiple bridge cases, and the three single BCA multiple algorithm cases above. In these cases, instead of a single bridge that does multiple algorithms, we have one BCA for each algorithm. This really means that there is a separate name for the bridge for each algorithm, but not necessarily a different physical CA workstation. The Fully Connected Multiple Bridge CA, illustrated in Figure 2 (a) corresponds directly to the fully connected multiple algorithm bridge of (1) above. It has exactly the same number of certificates and the same redundant certificates. Similarly, the multiple bridge CA, bridges issue mixed certificates case of Figure 2 (b) corresponds to the multiple algorithm bridge, bridge issues mixed certificates case of (2) above. The multiple bridge CA, PCAs issue mixed certificates case, illustrated in Figure 2 (c), corresponds to the multiple algorithm s bridge, PCAs issue mixed certificates case of (3) above. The only difference in these cases is whether the bridge CA gets one name, which it uses for all algorithms, or a different name for each algorithm it supports. Do the separate names help in certification path building? I suspect that probably they could, but only if there were some gen-

[1] This does not sound a particularly onerous requirement, but one suspects that there may be some CA products used in PCAs that might demand "proof of possession" before they issue a certificate, yet be unable to validate some of the BCA algorithms.

erally recognized convention about the names and keys that certification path building software could recognize.

Split Bridge CA Approach

In this approach, illustrated in Figure 2 (d), the Bridge CA is decomposed into as many Bridge CAs as there are signature algorithms. Each Bridge signs with only one of the algorithms and cross-certifies only with those PCAs that use the same algorithm, using consistent certificates. The Bridge CAs cross-certify among themselves, with mixed algorithm certificates. The only mixed algorithm certificates would be between the components of the Federal Bridge CA. If the composite Federal Bridge CA cross certifies with other Bridge CA's it would do as it does to PCAs that is with single algorithm certificates.

As always, the separate Bridge CAs are logically, but not necessarily physically, separate entities. A single CA workstation that could sign with each algorithm could be used to implement all of the BCAs. Each of the BCA components would, however, have its own name. Each BCA component would issue an ARL signed with its own algorithm.

The major disadvantage to this approach, as opposed to the multiple algorithm bridge, is that certification paths between users of different algorithms would have one extra certificate in the certifi-

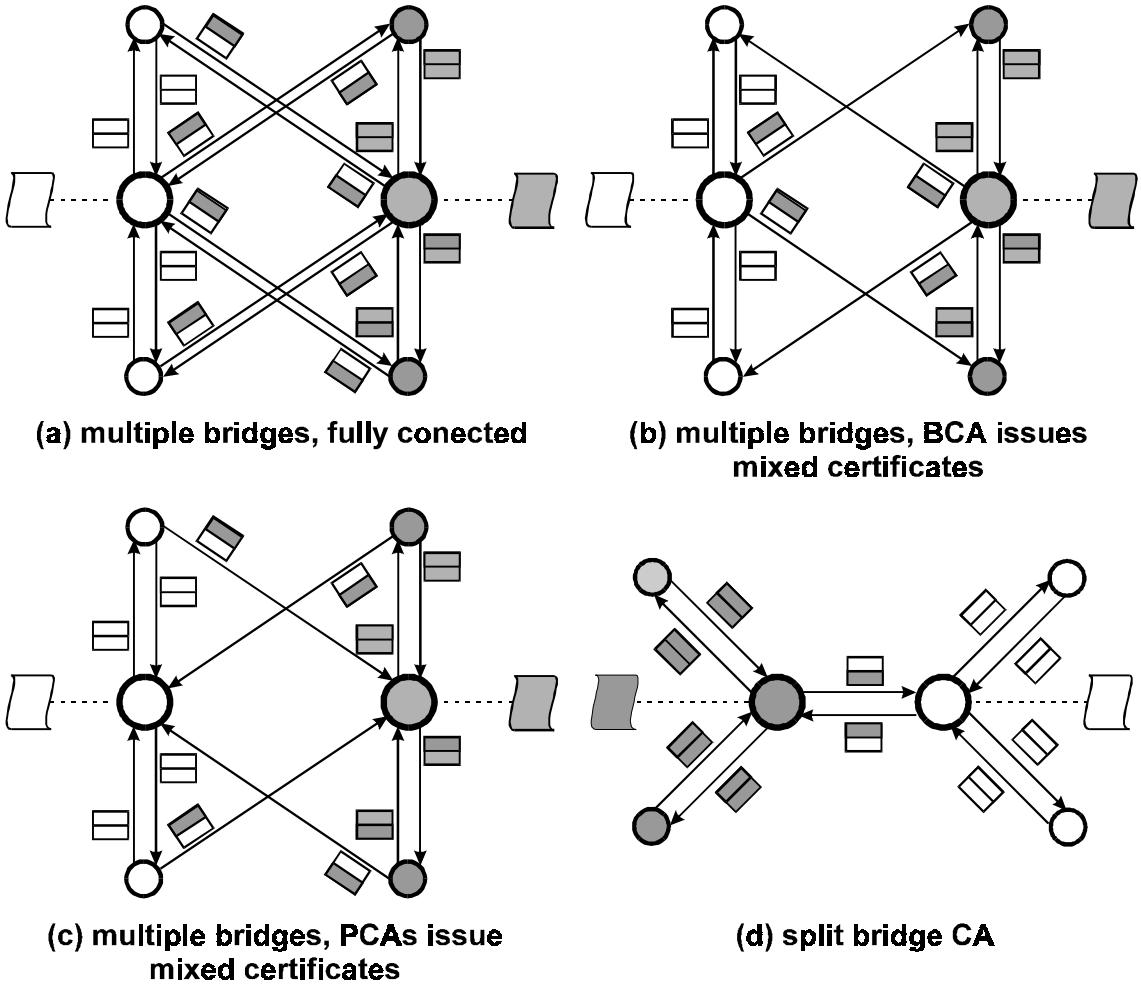

Figure 2 - Multiple Bridge Architectures

cation path. However the different names of the Split BCA components should simplify finding the needed hybrid certificates. Additionally, the number of hybrid certificates would be m × (m -1), which should be fewer than needed for the multi-algorithm BCA approach (since there should be many more PCAs than algorithms).

Furthermore, although I have not depicted it, there is a single CA multiple algorithm approach that is directly analogous. A multiple algorithm BCA could "self-certify" each of its keys with each other. I would worry, however, about the ability of clients to find and use these certificates to build certification paths.

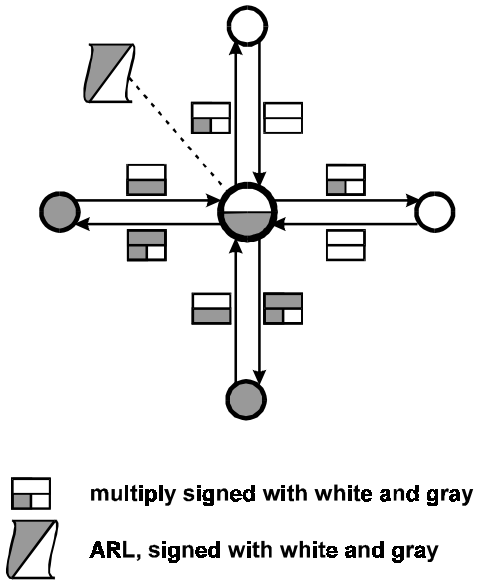

Figure 3 - Multiply Signed BCA Certificates

Multiply Signed BCA Certificates

Carlisle Adams has proposed a mechanism for constructing multiply signed (by different algorithms) documents, including certificates and CRLs, without changing the X.509 standard, by adding a new Algorithm ID for multiply signed. Appendix A summarizes the mechanics of this proposal. It then becomes possible to have the Bridge CA issue a PCA a single certificate signed by every algorithm supported by the BCA.

This approach has the considerable virtue that a single BCA certificate has the signatures to build a path with any algorithm supported by the bridge. This seems very efficient. But it does presuppose the introduction into all clients of a new, and rather different algorithm identifier, that supports multiple algorithms. This seems a huge assumption. However, this approach could possibly be combined with one of the other approaches, as a transition step until clients had widely implemented the new multiple signature OID. One question occurs to me: how does the Authority Key Identifier extension work in such a certificate?

Some Observations about Certification Path Building

I believe that it is conventional, at least in hierarchical systems, to build certification paths, starting at the signatory's certificate, and working "backwards" from there to find a path to the key the relying party trusts, usually the root's key. This is easy to do because the keys of subordinate CAs are all certified by the same parent. And, when a certificate is signed by one of several keys held by a CA, the issuer key ID field helps to find which one.

There is a doesn't seem to be any way to ask a directory to select certificates signed with a particular algorithm. But mixed algorithm BCA certificates will be combined into certificate pairs. In "fully connected" BCA architectures, both certificates of the pair would probably be single algorithm certificates, or both would be mixed algorithm certificates. In the cases where only the BCA, or only the PCAs issue mixed certificates, many certificate pairs would contain one mixed and one single algorithm certificate. So we should be able to get a directory to find certificate pairs where forward has one of the algorithms we need and reverse has the other.

There may be an advantage in the "fully connected" case, in that both certificates in a mixed algorithm certificate pair would be mixed certificates, more or less the mirror images of each other. If

we need to construct a bi-directional certification path, this would provide all the needed mixed certificates in one certificate pair.

Why would we need to construct a bi-directional certification path? Well, possibly for S/MIME. Alice wants to send a signed, encrypted message to Bob. She needs to validate the path to Bob's encryption certificate, while she also may wish to supply a certificate list from Bob's CA to Alice's certificate with her message. If both Alice and Bob were in mesh domains, the two certification paths would probably be mirror images of each other.

Table 1 - The Number of Certificates Needed for Each Approach

Case	BCA-PCA certs.	BCA-PCA Xcert pairs
Preferred Algorithm BCA	$2n$	$2n$
Multiple Algorithm. BCA, fully connected	$2n \times m$	$2n \times m$, *or* * $2n \times m^2$ **
Multiple Algorithm BCA, BCA issues mixed PCA Certs.	$n \times (m + 1)$	$2n \times m$
Multiple Algorithm BCA, PCA issues mixed BCA Courts.	$n \times (m + 1)$	$2n \times m$
Multiple. Bridges, fully connected	$2n \times m$	$2n \times m$
Multiple Bridges, BCA issues mixed PCA Certs.	$n \times (m + 1)$	$2n \times m$
Multiple Bridges, PCA issues mixed BCA Certs.	$n \times \times$	$2n \times m$
Split Bridge CA	$2\ n + m \times (m - 1)$	$2 \times (2\ n + m \times (m - 1))$
Multiply signed BCA Certificates	$2n$	$2n$

n is the number of PCAs and m is the number of BCA supported algorithms
* assumes "mismatched pairs" (e.g., gray certifies white in one direction, but white certifies gold in the other) aren't created.
** assumes all possible cross certificate pairs are created.

Conclusions

This paper has postulated that trust domains would use a single algorithm. It is sensible in the interest of local interoperability and efficiency to maintain homogeneous trust domains. However, it should be apparent that the same technique use with the bridge CA could be applied to a principal CA to support multiple algorithms within trust domains.

The mixed algorithm certificates needed to allow multi-algorithm operation in a PKI are logically provided by the Bridge CA in conjunction with the PCAs. There is no need for any CA except the Bridge CA to be able to sign with more than one algorithm.

Table 1 gives the number of PCA-BCA certificates and cross-certificates needed for each approach. The most efficient choice is to use the preferred algorithm approach, however it may be politically impracticable to select a preferred algorithm. The second most efficient would be the multiply signed certificate approach. That, however, assumes the pervasive introduction of a new and unusual algorithm identifier in clients.

The remaining choices are all workable, if less efficient. They do not force us to pick a single preferred algorithm or invent and propagate a new multiple signatures algorithm identifier in clients, and may therefor be preferable. In general, the remaining approaches can be divided into approaches where one BCA does multiple algorithms, and corresponding approaches where there are multiple BCAs that each individually does one algorithm. The two are not profoundly different, however certification path creation and practical implementation considerations may possibly favor one or the other. I have not been able to identify why that should be the case.
Similarly, there are "fully connected" schemes that populate all the mixed certificates that can be created, and schemes that populate only the minimum required to guarantee that needed certification paths exist. Again, I have not been able to identify a compelling advantage to any of the schemes. I speculate that if there is an advantage, it probably is because one arrangement or another is better suited to algorithms for finding certification paths.

References

[TWG-98-18] W. E. Burr and W. Poll, "A Federal PKI with Multiple Digital Signature Algorithms", April 8, 1998

[TWG-98-29] W. E. Burr, "Proposed Federal PKI Architecture," 19 May 1998

[TWG-98-31] Draft Federal PKI Concept of Operations, 3 June 1998.

[TWG-98-33] R. Guida, "Notional Description of a Federal Policy Management Authority and Bridge Certification Authority" June 1998

Appendix A - Multiple Signature Certificates

At the July 13 TWG meeting Carlisle Adams proposed that multiple signature certificates could be issued by the bridge CA to provide certification paths between a relying party who used one signature algorithm, and a signatory who used another. A certificate can be decomposed, or restated (in my pseudo ASN.1 notation) as:

```
Certificate     ::=     SEQUENCE {
        toBeSigned                      -- the body of the cert.
        SEQUENCE {
                algorithm               -- simply an OID
                parameters OPTIONAL } --a sequence that depends on the algorithm
        ENCRYPTED-HASH { ToBeSigned }}  -- a bit string, that also depends on the
                                                algorithm
```

In this case **toBeSigned** is the body of the certificate, but it doesn't really matter, it could be any message we want to sign, the technique isn't limited to certificates. Carlisle observes that the **algorithm** is just an OID, and we are free to introduce new OIDs into certificates, and several bodies have defined OIDs for different algorithms; indeed there is an overabundance of such OIDs. So we are not doing any violence to X.509, if we add another, call it **multipleAlgorithms**.

The interesting thing is that **parameters** and **ENCRYPTED-HASH** all entirely depend on the **algorithm OID**. Carlisle exploits this by proposing to define another algorithm OID, **multipleAlgorithms**. The associated **parameters** is a list of AlgorithmIdentifier (i.e., a list of {OID, Parameter} pairs), of several individual algorithms used to sign the certificate. And **ENCRYPTED-HASH,** which is a bit string, now becomes a sequence of bit strings (the whole sequence then encoded as a BIT STRING, to comply with traditional syntax), each one corresponding to a different signature on the certificate, in the same order as the AlgIds given in **parameters**.

This would create sort of super certificate to be issued by the BCA to PCA,s certifying the PCA's key with all the algorithms the BCA supports. Only one multiple certificate is needed per PCA.

If we have taken over parameters for this purpose, where do we get the parameters used to validate the signatures for DSA or ECDSA? This bothered me at first. It turns out that we should never get parameters from the field of a signature; that is insecure because the signature does not protect the parameters (and it would be circular if it did). We get the parameters we need to validate the signature from the **subjectPublicKeyInfo** of certificate the relying party's PCA issued to the BCA, just as we would for any other signature. It is important to understand that only one of the signatures, that for the algorithm used by the relying party's PCA, would be validated by relying party, when the multiple algorithm certificate is used.

Carlisle has not actually changed the ASN.1 definition of a certificate at all. He has simply added an new (somewhat unusual) signature algorithm, which is a list of other algorithms, but which everybody needs to process (at least for validation) if this is to work. Carlisle's approach has some advantages:

- It eliminates the complexity in finding involved in finding the "correct" one of several mixed algorithm certificates issued by the BCA to each PCA in my "multiple algorithm BCA" approach;

- It eliminates the complexity inherent in finding the "correct" one of several mixed algorithm certificates issued by each PCA to the PCA in Santosh's variation;
- As compared to the "split bridge" approach, it eliminates the complexity of splitting the Bridge into several CAs, and eliminates one additional certificate in each path.
- It eliminates the complexity of keeping multiple mixed algorithm certificates in synch (with respect to things like expiration) for any of the alternatives above;

The singular disadvantage of Carlisle's proposal is that it requires the adoption of a new, fairly unusual algorithm identifier by most of the client vendors, before it will work. Carlisle argues (correctly, I believe) that the actual complexity of this is much less than adding the ability to validate a second or third signature algorithm, which is required in any case. But it is extra, and perhaps a big assumption.

One factor that I need to think some more about is the general utility of this multiple signature technique. We often have multiple signatures on paper documents. Surely there are other applications for a multiply signed document. My intuition is that it may be hard to justify introducing this just for multiple algorithms and the bridge CA, which has other, if probably less desirable solutions, but it may be worthwhile if we can identify other important applications of the technique.